ASSIGNMENT - 2

1) $p = 0.25 \quad n = 1000$

The simulation has been done in the file Q1.py
To get the result, use this command in Terminal:

```
python3 Q1.py
```

The output comes out to be:

Probability of at least 240 A's in the sequence = 0.7784

Explanation of code:

For each simulation we generate a random sequence of $n = 1000$ numbers, as follows:

i) generate a random number from 0 to 1
ii) If value of this random number generated is less than p(0.25 in our case) then add a 1 to the sequence, otherwise 0

The sum of the sequence gives the number of A's in the sequence.

Counting the number of times we get a sum of 240 or more and then dividing it by the number of simulations, gives the answer.

Normal Approximation to Binomial Distribution

$p = 0.25$; $q = 1 - p = 0.75$; $n = 1000$; $\mu = n \times p = 250$

By central limit theorem, we can approximate the binomial distribution into a normal distribution.

$$\sigma = \sqrt{npq} = \sqrt{1000 \times 0.25 \times 0.75} = 13.69$$

$$z = \frac{240 - \mu}{\sigma} = \frac{240 - 250}{13.69} = -0.7305$$

On comparison, we see both the values are very close.

Using z-score table, we get z score corresponding to -0.7305 as 0.2327

$\therefore$ $P(x \geq 240) = 1 - P(x < 240) = 1 - 0.2327 = 0.7673$

2) $p = 0.3$ $n = 10$ $P(x=0)$, $P(x=2)$, $E(x)$, $Var(x)$

$p = 0.3$; $q = 1-p = 0.7$; $n = 10$

$P(X=x) = {}^{n}C_{x}\, p^{x} q^{n-x}$

$\therefore P(X=0) = {}^{10}C_{0}\, 0.3^{0}\, 0.7^{10} = 0.02824$

$P(X=2) = {}^{10}C_{2}\, 0.3^{2}\, 0.7^{8} = \frac{10\times9}{2} \times 0.3^{2} \times 0.7^{8} = 0.23347$

$E(X) = n \times p = 10 \times 0.3 = 3$

$Var(X) = n \times p \times q = 10 \times 0.3 \times 0.7 = 2.1$

3) Applications of k-mer analysis :

k-mers with $k \geq 2$ are used to identify regions with aberrant base compositions that show genome segments obtained by lateral transfer.

The different applications are as follows :

i) For eukaryotes, gene regions have different base composition compared to non-genic regions. Various gene classes have different codon usage frequencies that differs between organisms. Hence, they can be used to identify horizontally transferred genes.

ii) Observed frequencies of k-words can be used to analyze DNA sequences.

iii) k-tuple ($k > 3$) frequencies may also be useful in predicting whether an unannotated sequence is coding or non-coding.

iv) k-mer distributions are well preserved among related strains/species. Therefore, bacterial genomes can be clustered into natural groups on the basis of k-mer distribution similarities.

4) The two sequences are :

GGCTGCAACTAGCTC

GGGTAAGCTTGC

	G	G	G	T	A	A	G	C	T	T	G	C
G	X	X	X				X				X	
G	X	X	X				X				X	
C								X				X
T				X					X	X		
G	X	X	X				X				X	
C								X				X
A					X	X						
A					X	X						
C								X				X
T				X					X	X		
A					X	X						
G	X	X	X				X				X	
C								X				X
T				X					X	X		
C								X				X

The colored crosses denote a match. The conserved regions of length > 2 are:

i) AGCT (sequence 1: 11 to 14, sequence 2: 6 to 9 (included)) : in violet

ii) GCT (sequence 1: 2 to 4, sequence 2: 7 to 9 (included)) : in orange

iii) TGC sequence 1: 4 to 6, sequence 2: 9 to 11 (included)) : in green

---

Pseudocode for Q1:

specify p and no. of simulations

count of 240A's initialize to 0

for a → 0 to (no. of simulations - 1) :

X = random array of length 'n'

If any element in X < p, then X[at that index] = 1.

Add all elements.

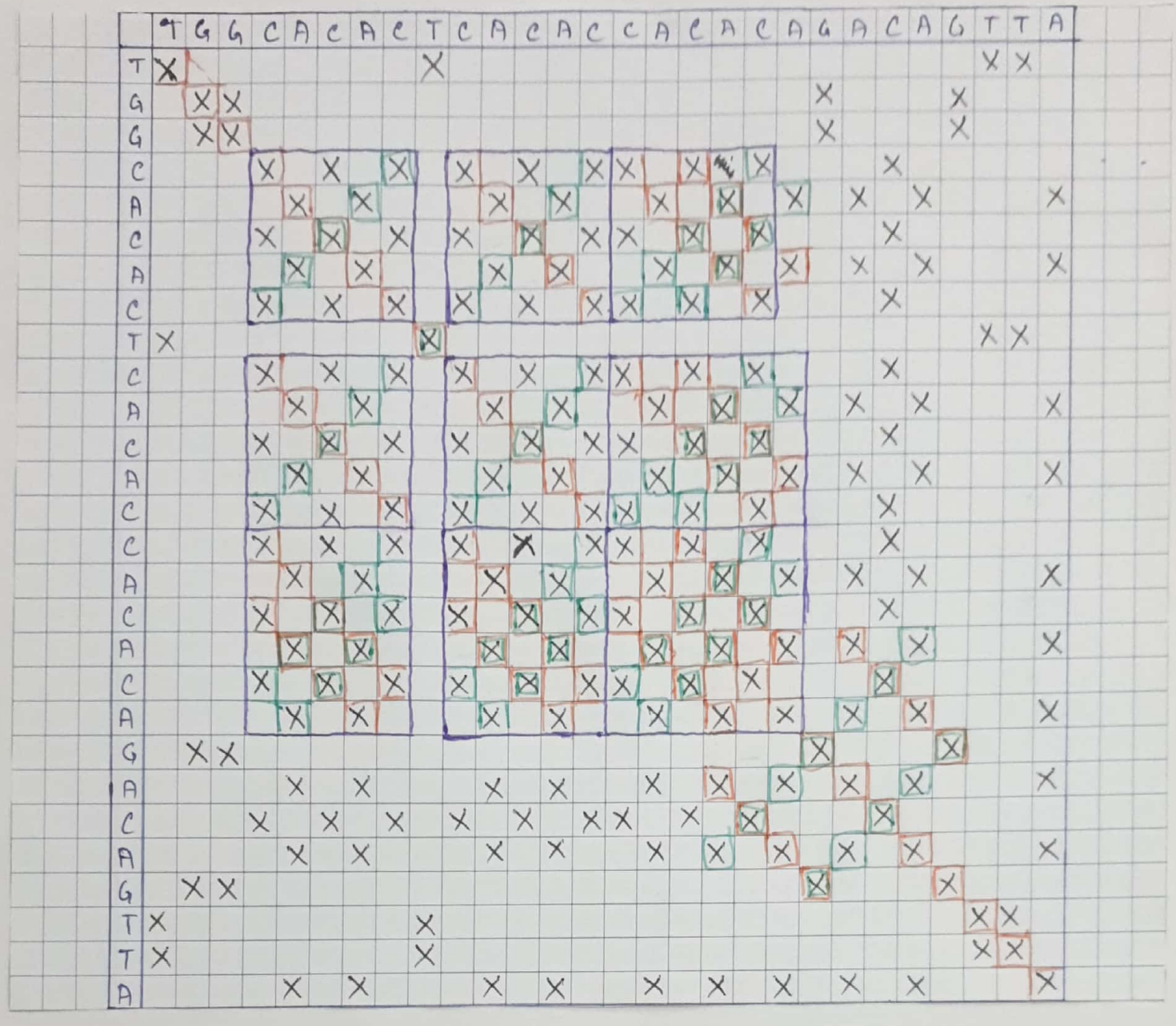
If (sum ≥ 240), then - count of 240A's += 1

(After for loop)

Probability = count of 240A's / no. of simulations

5) Identifying repeat region:

TGGCACACTCACACCACACAGACAGTTA



Mainly two kinds of repeats:

1) Low complexity regions in violet

They can be identified by horizontal/verhical rows of dots/crosses that can merge into rectangular or square patterns

2) Internal repeat regions

(i) Forward repeats in orange

They can be identified as crosses(X) or dots(.) trying to form a forward diagonal (top left – bottom right)

(ii) Inverted repeats in green

They can be identified as crosses(X) or dots(.) trying to form a backward diagonal (bottom-left – top right)

Pseudocode for Q5: matches marked as "x"

```
def show-dotplot(seq1, seq2):          # This code gives a string of
    dotplot = "| |"                    # the whole table as it should
    for j -> 0 to length(L2)-1:        # be shown
        dotplot += L2[j] + '|'         # L1, L2 are lists formed from
    dotplot += "\n|"
    for j -> 0 to length(L2)-1:        # seq1, seq2 respectively as
        dotplot += '-|'                # they were strings
    dotplot += '-|\n'
    for i -> 0 to length(L1)-1:        # Checking matches
        dotplot += '|' + L1[i] + '|'   # for each element
        for j -> 0 to length(L2)-1:
            val = "x" if (L1[i] = L2[j]) else " "   # of both (list)
            dotplot += val + '|'       # sequences
        dotplot += '\n'
    return dotplot
```

Function name is show-plot which takes two string as input

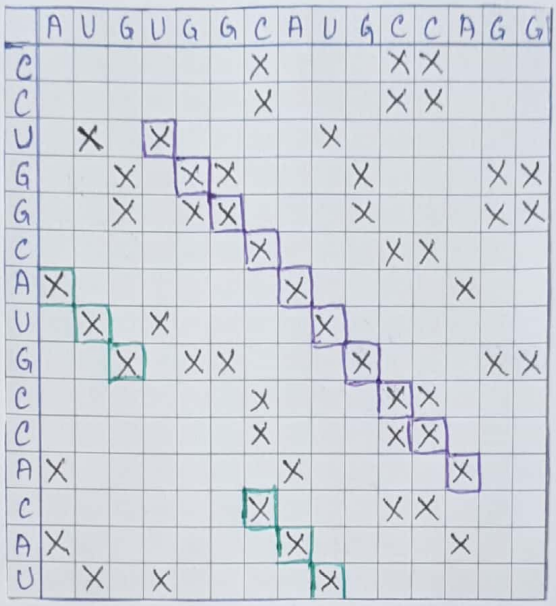
Main modules:

```
seq = "TGGCACACTCACACCACACAGACAGTTA"   # given in question
print (show-dotplot(seq, seq))         # since self-matching.
```

6) First sequence is listed in 5' to 3' direction in the horizontal direction and its complementary sequence is listed along the vertical direction also in the 5' to 3' direction.

Matrix then tracks the identical matches.

Self-complementary regions can be seen as diagonals from top left to lower right



Sequence:

AUGUGGCAUGCCAGG

Complementary Sequence:

CCUGGCAUGCCACAU

Longest self-complementary regions found here is shown in VIOLET (length = 10) and is: UGGCAUGCCA

Other small regions marked in green: AUG, CAU

Dotplots are also shown in the form of code for questions 4, 5, 6. The codes (files in '.py') are Q4.py, Q5.py and Q6.py respectively.